Attorney Docket No. 35581.00.0003

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

FILING OF A UNITED STATES PATENT APPLICATION

Title:

APPARATUS AND METHOD FOR PRECLUDING E-MAIL DISTRIBUTION

Inventors:

Bronwyn Syiek 45 Southern Heights Avenue San Francisco, California 94107	Nina Bhanap 960 Roble Avenue, Apt. B Menlo Park, California 94025
Rakesh Gowda 4021 Fair Oaks Avenue Menlo Park, California 94025	Ramesh Guntha 3803 Darwin Drive, Apt. 258 Fremont, California 94555

Attorney of Record
Robert S. Beiser, Reg. No. 28,687
Vedder, Price, Kaufman & Kammholz
222 North LaSalle Street, Suite 2600
Chicago, Illinois 60601
Telephone (312) 609-7500
Facsimile (312) 609-5005

Express Mail Label No.: EV 320527165 US

Date of Deposit: June 20, 2003

I hereby certify that this paper is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. § 1.10 on the "Date of Deposit," indicated above, and is addressed to the Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Name of Depositor: Carmen M. Camarena
(print or type)

Signature: Carmen M. Camarena

APPARATUS AND METHOD FOR PRECLUDING E-MAIL DISTRIBUTION

Field of the Invention

[0001] This invention is directed to a method and apparatus for managing e-mail message transmission. More specifically, the present invention is directed to a method and apparatus by which proprietary e-mail addresses can be distributed to third party e-mail address filtration service providers.

Background of the Invention

[0002] Electronic mail, which is also known as "e-mail," is the transmission of messages over communication networks, such as the Internet. E-mail messages can take the form of notes that are entered into a computer using a keyboard. They can also take the form of attachments to messages.

[0003] It is well known that e-mail has many advantages over other forms of communication. E-mail is much less expensive than mail service provided by the U.S. Postal Service (U.S.P.S.). Unlike the U.S.P.S. mail delivery, e-mail is delivered almost instantaneously. When an e-mail message cannot be delivered, an error notification is also received almost instantaneously. Direct solicitations by telemarketing is slow and expensive. Because commercial messages via e-mail are inexpensive to both create and deliver, and because they are delivered almost instantaneously, e-mail is quickly become a marketing tool of choice for many companies.

[0004] Unfortunately, the advantages of commercial message transmission by e-mail have caused such messages to overload networks that deliver them. Many Internet service providers claim that unsolicited commercial messages, which are known as "spam" account for most of the e-mail messages they deliver.

[0005] SPAM® is a registered trademark of Hormel Foods, Corporation. For years, SPAM® has been used to identify Hormel's deviled luncheon meat spread. Perhaps unfortunately for Hormel, the word "spam" is now also being widely used to refer to unsolicited commercial e-mail messages sent to recipients who have not indicated a willingness to receive such unsolicited email messages from the sender. Spam now taxes the capabilities of many Internet service providers and is widely considered to be an annoyance by Internet users. Legislation to prohibit spam might not be far off.

[0006] Many entities that send commercial messages by e-mail have begun to attempt to control unwanted e-mail message transmissions. Because many e-mail advertisers use e-mail lists that are provided by mass mailers, filtering e-mail address lists of an e-mail mass mailer to control spam is problematic because e-mail mass mailers are reluctant to turn over trade secrets embodied in e-mail address lists. Consequently, controlling the transmission of unwanted e-mail distributed by third-party service providers is inhibited by trade secret concerns. A method and apparatus by which e-mail address lists could be filtered to remove addressees who do not want to receive unwanted e-mail messages would be an improvement over the prior art.

A Brief Description of the Drawings

[0007] Fig 1 is a depiction of the operation of a hash coder, implemented in either hardware or software, but known in the prior art.

[0008] Fig. 2 is a block diagram of a system for filtering e-mail addresses prior to e-mail transmission.

[0009] Fig. 3 is a flow diagram of a method for controlling electronic mail message transmissions.

[0010] Fig. 4 is a flow diagram of a method used in the method of Fig 3 showing an element-by-element comparison of hash codes.

Detailed Description of the Preferred Embodiments

[0011] A hash function is any mathematical algorithm that converts a variable-length string of letters and/or numbers into a fixed-length output string of letters and/or numbers. The output of a hash function is known as a "hash value" or "hash code." Among other things, hash functions are used to create digital signatures of strings of characters.

[0012] As is well known, an e-mail address has at least three parts: a name or other designation of an addressee followed by a domain name and a domain. The name and domain name are usually separated by the "@" symbol.

[0013] E-mail addresses in general take the form of:

[0014] <name of addressee>@<domain name>.<domain>

[0015] Where <name of addressee> is an message recipient identifier.

[0016] Examples of e-mail addresses (all of which are fictitious) include:

[0017] john.smith@xyz.com

[0018] fred_morgan@abc.gov

[0019] s.person@internetserviceprovider.com

[0020] A hash coder is considered to be a device or software-implementation of a hash function. A hash coder processes variable-length input strings, mathematically operates on the characters of an input string and generates a fixed-length output number or string that is uniquely-valued according to the input string. For a given hashing algorithm, a particular input string will always be encoded into the same output string. Stated alternatively, for every string input to a particular hashing function, there will be a unique output string. Hash codes therefore provide a way to encode an e-mail address string in a way that the hash code can uniquely identify the address, without disclosing the address from which it was calculated. De-hashing a hash code is virtually impossible.

[0021] Fig. 1 depicts a hash coder 100. As is known in the art, the hash coder can encode variable-length strings of text (ASCII representations of alpha-numeric characters) that are e-mail addresses, into fixed-length strings of alpha-numeric characters. Examples of output hash codes are identified by reference numerals 104, 108, 112, and 116 and are depicted as being output from the hash coder 100.

[0022] Hash coder 100, can be implemented in hardware using combinational and/or sequential logic devices. Hash coder 100 can also be implemented using software or computer program instructions that are executed on an appropriately-capable processor, examples of which include a microprocessor or micro controller as well as main-frame computer.

[0023] In Fig. 1, variable length input e-mail address strings, identified by reference numerals 102, 106, 110, and 114, are input to the hash coder 100. The characters that make up each input string 102, 106, 110 and 114 are operated on by the hash coder 100 and are mathematically processed to be converted by the hash coder 100 into multi-character, fixed-length string of letters and/or digits. Each *output* string 104, 108, 112, and 116 represents an input e-mail address and is considered to be a "hash code." Each hash code corresponds to, and therefore uniquely identifies a string of characters, which is the e-mail address from which the hash code was generated and to which a message would be sent. The hash coder can accept as input, lists of e-mail addresses and generate lists of hash codes, each of which uniquely represents an e-mail address.

[0024] As used herein, a "list" is a multi-element data structure that has a linear organization (first, second, third, . . .) but that allows elements to be added or removed in any order. A list of hash codes is a multi-element data structure of hash codes.

[0025] Those of ordinary skill in the art know that de-hashing a hash code to recover a string from which the hash code was calculated is virtually impossible because of the computational complexity of most hash code algorithms. Hash codes of e-mail addresses can therefore be distributed to third parties without disclosing the actual addresses from which the hash codes were generated. In other words, hash coding is a way of encoding addresses so as to preserve their identity.

[0026] In converting an e-mail address to a hash code, the alphabetic characters of an e-mail address are first mapped or converted to lower case characters. (When hashed, an upper case letter and the same lower case letter will yield different hash codes. The characters of the address are then input to a hash coder, which generates an output hash code.

[0027] Inasmuch as e-mail addresses are being converted into a form from which the actual e-mail address is not readily apparent, the e-mail addresses are considered herein as being "encoded." Those of skill in the art will recognize that in addition "encoding" e-mail addresses by hash coding, other processes that convert variable-length strings to a fixed-length string could be used instead of hash coding. Therefore, as used herein, the term "encoded" should be construed to include both hash coding *per se*, but also to include any other process by which variable length strings can be converted to a fixed-length string.

[0028] Fig. 2 depicts the components of a system 200 for controlling electronic mail message transmission over a data network, such as the Internet. In one embodiment of a method depicted by Fig. 2, a merchant 203 (of goods and/or services) selectively distributes its commercial/promotional message (advertising its goods and/or services) via e-mail using e-mail address filtering services of an e-mail list manager 210. In another embodiment, the merchant can also use the services of an e-mail mass mailer 212.

[0029] As used herein, the term "merchant" refers to a commercial entity that is identified in Fig. 2 by reference numeral 203. A "merchant" distributes commercial messages over the Internet but in order to do so, such an entity requires, uses or operates one or more computers, which are identified in Fig. 2 by reference numeral 202. The "merchant" is a "client" to the list manager 210, described more fully below.

[0030] As used herein, an "e-mail mass mailer" is considered an entity that ordinarily distributes or sends e-mail messages indiscriminately, i.e., without any filtering or selectivity. An e-mail mass mailer's computer is shown in Fig. 2 and identified by reference numeral 212. For simplicity of illustration, the e-mail mass mailer's computer 212 and the entity are considered to be one and the same.

[0031] Among other things, in one embodiment, the list manager 210, which is also an entity (not shown) that uses or operates one or more computers, provides a marketing function for the merchant. Such marketing activities can include creating commercial message content for the merchant 203. However, the list manager also provides a second service of receiving hashed e-mail addresses from the e-mail mass mailer 212 and filtering from the hashed e-mail addresses it receives, e-mail addresses to which a commercial message should not be sent. Because the hashed e-mail addresses provided by the mass mailer 212 are usually not sorted or filtered by the e-mail mass mailer, when the list manager 210 receives the hashed addresses, the list manager 210 sorts and filters the hashed addresses to remove hashed addresses to which an e-mail message should not be sent.

[0032] Part of the list manager's 210 services to the merchant 203 can include the provision of one or more web sites (not shown for clarity) that deliver advertising content on behalf of the merchant 203. Such web sites preferably solicit prospective customers of the merchant 203 to provide their contact information by which the prospective customer of the merchant 203 can be contacted. The information requested of the prospective customer on the web sites preferably includes an e-mail address to which information can be sent to the prospective customer via e-mail. Obtaining contact information of prospective customers through the use of web site advertising is more effective if prospective customers are directed to a web site that promotes the merchant's goods and/or services.

[0033] One way to refer prospective customers to commercial messages at a web site is by way of a broadcast e-mail message (i.e., spam) directing someone to visit a web site whereat the merchant's 203 goods and/or services are advertised. If a person visits a web site after receipt of a suggestive e-mail, and then provides his or her contact information, such a person has identified themselves as being interested in the goods or services of the merchant and should be considered for follow-up marketing. In another embodiment of the invention, an unsolicited commercial e-mail message is delivered to numerous addressees having prior business relationships with the e-mail sender directing the e-mail recipients to one or more of the list manager's 210 web sites whereat the merchant's business is promoted. Visitors to a web site and who provide their contact information are considered to be leads that the merchant 203 can follow up with to stimulate additional business.

[0034] In promoting goods and services using e-mail, many merchants prefer to avoid offending prospective customers by sending them unsolicited messages, redundant or duplicative messages, or sending messages to people who have specifically requested to not be sent unsolicited commercial messages. Accordingly, commercial messages of the merchant 203 should not be sent to various e-mail addresses and are herein referred to as e-mail addresses to which a message should not be sent. Such e-mail addresses can be obtained from at least three different sources.

[0035] One source of e-mail addresses to which an advertisement or promotion should not be sent, are e-mail addresses known to a merchant 203 to be inappropriate. Examples of such addresses include addresses of existing customers to whom additional promotional material or solicitations of new business would be inappropriate. In addition to existing customers, in some instances, sending solicitations for new business to former clients or customers who no longer need the goods or services of the merchant would be inappropriate. Individuals who have asked to be excluded from receiving unsolicited commercial messages should also not be sent an unsolicited commercial e-mail.

[0036] Another source of e-mail addresses to which an advertisement or promotion should not be sent are addresses that are known as such to the list manager 210. Examples of such addresses include the addresses of people who have asked the list manager 210 to exclude them from receiving unsolicited e-mail messages regarding that particular merchant only. A request to be excluded from receiving unsolicited e-mail can come to the list manager in the form of a reply to a previously sent e-mail message (i.e., an unsubscribe request) or one of usually two links on an e-mail, on one of which a user "clicks" so as to unsubscribe from either mass mailer's data base by "clicking" on link or unsubscribing from the list manager's 210 data base by "clicking" on the other link. The list manager 210 can receive specific requests from persons who ask to be excluded from all unsolicited commercial message deliveries but the list manager 210 can also purchase lists of addresses to which unsolicited messages regarding any merchant should not be sent. Addresses to which no messages should be sent are considered herein to be a "black list." Sending an unsolicited commercial e-mail message can be counterproductive. Therefore, lists of possibly millions addresses to which a commercial message will be sent, such as lists from a mass mailer, are first compared to addresses known to be addresses to which an unsolicited commercial message should *not* be sent. Addresses to which unsolicited messages should not be sent are purged from an extended list of addresses to avoid sending what will be considered to be "spam."

[0037] In one alternate embodiment, the list manager 210 acts on behalf of the merchant 203 and generates a commercial message, which the list manager 210 sends itself to e-mail addresses or forward to an e-mail mass mailer 212 for delivery. In another alternate embodiment, the list manager 210 creates a commercial message for the merchant 203 but sends the message to an e-mail mass mailer 212 for delivery. In yet another embodiment, marketing services can be provided by an entirely different entity not shown in the figures while the e-mail address list filtration is performed by the link manager 210.

[0038] In a preferred embodiment, the list manager 210 receives from the merchant's computer 202, e-mail addresses known to the merchant 203 to be addresses to which the list manager 210 should *not* send a commercial message on behalf of the merchant 203. E-mail addresses that are known to the merchant 203 to which a message should *not* be sent are purged by the list manager 210 from lists of e-mail addresses to which such messages will otherwise be sent.

[0039] The list manager 210 can also obtain from an e-mail mass mailer an extended list of e-mail addresses, to which a commercial message can be sent, i.e., a first list, albeit without any knowledge of whether any of the addressees wants or does not want to receive an unsolicited commercial message. This so-called "first list" of addresses of potential recipients of a commercial message can be obtained from an e-mail mass mailer 212 or from the aforementioned web sites. Regardless of where or how the "first list" of addresses is obtained, the first list might contain addresses to which a message should not be sent.

[0040] In an alternate embodiment, the list manager 210 can send to the e-mail mass mailer 212 one or more e-mail domains to which no messages should be sent. In such an embodiment, the e-mail mass mailer 212 purges from its list, all e-mail addresses that contain a particular domain name or any of the domain names specified by the list manager 210. Remaining e-mail addresses that do not have the objected-to domain names can then be hash coded into LIST-1, with LIST-1 being processed as described otherwise. The extraction of certain e-mail addresses having certain domain names by the e-mail mass mailer 212 can be accomplished a number of ways, including by way of additional functionality that accompanies the hash code software that is distributed to the e-mail mass mailer 212 from the list manager 210.

[0041] In a preferred embodiment, the e-mail addresses known to the merchant 203 as addresses to which a message should not be sent comprise a list that is known as a "suppression list" 204. Addresses in a suppression list are often the e-mail addresses of existing customers to whom another promotional message might prove aggravating. Suppression list entries also include former customers or clients, or entities/persons to whom a commercial message might be inappropriate in light of its subject matter.

[0042] Addresses lists that are known to the list manager 210 as addresses to which a message should not be sent are known as "unsubscribe lists" and "black lists." An "unsubscribe list" is considered to be a list of e-mail addresses that are of addressees who have requested the list manager 210 (or some other entity) exclude them from future e-mail mailings, albeit with respect to a particular merchant 203 (i.e., a particular client of the list manager 210). The "black list" is considered to be a list of addresses of addressees who have requested the list manager 210 (or some other entity) to exclude them from *all* future mailings for all merchants (i.e., all clients of the list manager 210) and who should not receive any unsolicited commercial messages.

[0043] In order to control the distribution of commercial messages and avoid sending them to addresses where such message are either unwanted or inappropriate, the list manager 210 purges from any list of addresses to which a commercial message will otherwise be sent, all addresses that are on either the suppression list 204, the unsubscribe or the black list 214. The result of purging suppression list 204 addresses and unsubscribe and black list addresses 214 from the first list is a fourth list of addresses to which a commercial message may be sent.

[0044] As is well known, customer lists represent trade secrets. E-mail addresses lists are also generally considered customer lists. Exchanging lists of e-mail addresses between the merchant 203, the list manager 210 and the e-mail mass mailer 212 is therefore problematic. By using the invention disclosed and claimed herein, exchanging e-mail addresses without disclosing trade secrets is enabled by hashing the e-mail addresses prior to delivery to another entity. In a preferred embodiment, e-mail addresses to be exchanged between a merchant 203 or its computer 202, a list manager 210 and/or an e-mail mass mailer, are coded by hash coding. By hash coding the variable-length strings of e-mail addresses, the addresses are converted to fixed-length strings or letters and/or numbers, each of which uniquely identifies an e-mail address. Because the hash codes are all the same length, testing them for identity is greatly simplified. By converting lists of e-mail addresses into corresponding lists of hash codes, hash codes on the second and third lists can be purged from lists of hash codes that were computed from e-mail addresses of the first list to yield a list of hash codes that identify addresses to which an e-mail message may be sent. Hashing the e-mail addresses into hash codes enables the existence of an e-mail address in one list to be determined to be in another list without disclosing the actual addresses in any of the lists.

[0045] In order for the method disclosed herein to work, hash codes that are sorted and filtered need to be computed using the same hash function. As a result, and as shown in Fig. 2, in a first step, hash code software that implements a hash code algorithm is provided to (the computer of) the merchant 202 and to the e-mail mass mailer 212, thereby providing them with the ability to hash e-mail addresses into uniform-length hash codes.

[0046] In addition to distributing the hash code software, in step 218, the software that the list manager 210 distributes to the e-mail mass mailer 212 performs additional functionality on the computer(s) of the e-mail mass mailer 212. In a preferred embodiment, the software distributed to an e-mail mass mailer 212 from the list manager 210 also performs the functions of comparing the lists returned to the e-mail mass mailer 212 by the list manager 210 to the lists originally sent out by the e-mail mass mailer. In addition and as set forth below, software distributed by the list manager 210 to the e-mail mass mailer also provides functionality by which one or more e-mail address domains can be identified to the e-mail mass mailer 212. The e-mail mass mailer 212 can then delete, remove or exclude from the list of e-mail addresses that it will hash, all addresses within one or more domain names. In a preferred embodiment, the software distributed in step 218 also performs the function of extracting e-mail address strings from a hash code returned to the e-mail mass mailer, which is a procedure/process described more fully below.

[0047] After distributing the hash code software, in step 218 the list manager 210 receives a first list of hash codes (i.e., LIST-1) from an e-mail mass mailer 212, each of which represents an e-mail address. In step 208, the list manager 210 also receives a second list of hash codes from the merchant's computer 202 (i.e., LIST-2) each of which represents an e-mail address on the merchant's 203 suppression list 204. E-mail addresses in the list manager's 210 unsubscribe list and black list are also hash coded by the list manager 210 using the same hash code algorithm that was distributed to the merchant 202 and the mass mailer 212 to form a third list (i.e., LIST-3).

[0048] In order to expedite the search for hash codes in the various lists, the first, second and third lists of hash codes are each sorted by the list manager 210 using a prior art sorting algorithm known as "Quick Sort." Sorting techniques are well known. Other sorting algorithms could certainly be used. In a preferred embodiment, list entries are sorted in ascending order. However, list entries could also be sorted in descending order.

[0049] In step 220, sorted list entries are compared to each other to identify hash codes in LIST-1 that also appear in either LIST-2 or LIST-3. Hash codes in LIST-1 that appear in either LIST-2 or LIST-3 are purged from LIST-1 to yield a fourth list of hash codes, each of which represents an address to which an e-mail message may be sent (i.e., LIST-4). In step 221, LIST-4 is returned to the e-mail mass mailer (i.e., its owner), which extracts from the fourth lists of hash codes, each e-mail address on the first list, the hash code of which remains in the fourth list. The resultant list of e-mail addresses are considered to be a list of targeted addresses.

[0050] In step 221, the fourth list, which is also considered to be a "filtered" or "purged" list of hash codes, represents a set of addresses to which an e-mail message of the merchant 203 may be sent. After the filtered list is returned to the e-mail mass mailer in step 221, in step 222, the e-mail mass mailer 212 extracts or correlates each hash code in LIST-4 to the list of e-mail addresses from which LIST-1 was created to yield a set of targeted addresses to which a message can be transmitted in step 224

[0051] As shown in Fig. 2, the merchant's computer 202 is operatively coupled to and thereby accesses a mass storage device 206 on which is stored the suppression lists 204. The mass storage device 206 also stores program instructions, which imbue the computer 202 with the ability to access a data network such as the Internet via a data network interface (not shown for clarity of Fig. 2 but well-known to those of ordinary skill). By way of data transmissions over a network such as the Internet, the computer 202 can upload and download files via a data network, such as the Internet.

[0052] Program instructions stored in the mass storage device 206 include program instructions obtained from the e-mail list manager, which when executed by the computer 202 cause it to hash code e-mail addresses in the suppression lists 204 are stored in a file or database.

[0053] In the aforementioned preferred embodiment, hash codes corresponding to e-mail addresses of the suppression list 204 are uploaded via the Internet (or other data network) to the e-mail list manager 210. As shown in Fig. 2, the list of hash codes representing addresses to which an e-mail should not be sent is identified as "LIST-2" and sent to the e-mail list manager.

[0054] Using the same hash code algorithm that was provided to the commercial advertiser's computer 202, the e-mail mass mailer hash codes all of the addresses in the list(s) to which a commercial message might be sent. As shown in Fig. 2, the list of hash codes from the e-mail mass mailer is sent to the e-mail list manager and identified as "LIST-1."

[0055] Inasmuch as the commercial advertiser's computer 202 and the e-mail mass mailer 212 use the same hash code algorithm, the hash codes that are returned to the e-mail list manager 210 will be of the same length, which is a necessity in order to meaningfully compare hash codes in the various lists. If an e-mail address in the aforementioned suppression list (i.e., LIST-2) is also in e-mail mass mailer's list (i.e., LIST-1), when that particular e-mail address is hashed it should be converted into the exact same hash codes, regardless of who executes the hashing algorithm. That the hash codes are calculated using the same hash code algorithm enables the hash codes to be compared with any duplicates in the lists identifying duplicate addresses.

[0056] Fig. 3 depicts a method of controlling electronic mail transmission over a network using the system depicted in Fig. 2. As shown in the Fig. 3 legend, program steps outlined with solid lines are steps performed by the e-mail list manager 210 of Fig. 2; program steps outlined with dashed lines are performed by either the e-mail mass mailer 212 or the merchant's computer 202, which are also shown in Fig. 2.

[0057] In step 302, the e-mail list manager (210) pre-distributes hash coding software to both the client/commercial advertiser computer (202) and to the e-mail mass mailer (212). By using the same hash code algorithm, hash codes calculated by the merchant 203, the list manager 210 and the e-mail mass mailer 212 can all be meaningfully compared. The presence of the same hash code in LIST-1 and either LIST-2 or LIST-3 means that the same address in the first list of e-mails addresses (that was reduced to a particular hash code) must also be in the lists of e-mail addresses that were hashed to form LIST-2 and LIST-3, which are the lists of e-mail addresses to which an e-mail message should not be sent.

[0058] In step 304, the merchant's computer 202 hash codes the list of addresses in the aforementioned suppression list using the software provided by the e-mail list manager in step 302. In addition, in step 304, the merchant's computer (202) uploads the hash codes that represent e-mail addresses in the merchant's suppression list (204), to the e-mail list manager 210. The hashed addresses from the merchant's 203 suppression list comprise LIST-2.

[0059] Step 306 shows the e-mail list manager's 210 receipt of LIST-2 from the merchant's computer 202. In step 308, the e-mail mass mailer (212) hash codes its list of e-mail addresses and uploads the resultant list of hash codes, which is denominated as LIST-1, to the e-mail list manager 210. In an alternate and equivalent embodiment, the e-mail mass mailer (212) can hash its list of e-mail addresses at any time after it receives the hash code software or hash code algorithm. The e-mail mass mailer 212 does not need to wait until steps 304 or 306 have been executed in order to hash its e-mail addresses and/or send them to the list manager 210.

[0060] Step 312 shows the list manager's 210 receipt of LIST-1, which as depicted in Fig. 3 occurs after LIST-2 is delivered to the list manager 210 in step 304. The order of hash code list deliveries to the list manager 210 is not critical. In an alternate and equivalent embodiment, LIST-1 is delivered to the list manager 210 prior to, or simultaneously with the delivery of LIST-2 to the list manager 210.

[0061] In a preferred embodiment, the LIST-2 (i.e., the suppression list) is delivered to the list manager on a regular basis, e.g., whenever the merchant 203/client gets new e-mail addresses. The merchant 203 can either append to or replace a previously-sent suppression list. In "appending" e-mail addresses to a previously-sent list, the new e-mail addresses sent by the client are added to an existing suppression list. In "replacing" e-mail addresses, all of the e-mail addresses in the existing suppression list are deleted and new e-mail addresses are added to the suppression list. The e-mail mass mailer 212 sends its list whenever it is required to send a mass e-mail campaign. The list manager 210 need not receive the hashed email addresses in any particular order from either the merchant 203 or the e-mail mass mailer 212.)

[0062] As stated above, the e-mail list manager 210 can also generate it's own list of e-mail addresses to which e-mail messages should not be sent. Such addresses are referred to herein as an "unsubscribe list." The list manager 210 can also purchase lists of e-mail addresses to which messages should not be sent. Such addresses are referred to herein as "black lists."

[0063] "Black lists" and "unsubscribe lists" are both lists of addresses to which unsolicited commercial messages should not be sent. For simplicity of illustration and explanation, both the unsubscribe list and the black list are referred to herein as LIST-3. In step

318, the e-mail list manager 210 generates, compiles, acquires or otherwise obtains one or more of the e-mail addresses, which when hashed make up LIST-3.

[0064] As is well–known, an e-mail mass mailer 212 can collect millions of e-mail addresses, which when hashed will yield millions of hash codes. Determining whether an entry in LIST-2 or LIST-3 is in LIST-1 is accomplished in significantly less time if all of the lists are sorted prior to comparison. In a preferred embodiment, LIST-1, LIST-2 and LISTS 3 are sorted in step 320 in ascending alpha-numeric order using “Quick Sort” which is a sorting algorithm well-known to those of ordinary skill in the art. Quick Sort sorts the hash code lists in an ascending alpha-numeric order. The numbers zero through 9 are sorted first, then alphabetic characters from A to Z are sorted next. After LIST-1, LIST-2 and LIST-3 are sorted in step 320, their respective elements (i.e., the hash codes in each list) will be in an ascending alpha-numerical order.

[0065] In step 322, the entries of LIST-2 and LIST-3 that appear in LIST-1 are “purged” from LIST-1 to form LIST-4. Although “purging” elements from LIST-1 connotes a removal of elements, step 322 encompasses any methodology by which the entries of LIST-1 that do not also occur in LIST-2 or LIST-3 become components of LIST-4. LIST-4 could also be generated by flagging or marking elements of LIST-1 that also appear in LIST-2 or LIST-3. Regardless of how it is created, the hash codings of addresses of LIST-1 that do not appear on either LIST-2 or LIST-3 are sent back to the e-mail mass mailer 212. In a preferred embodiment, those addresses are sent as a list denominated as LIST-4.

[0066] Inasmuch as LIST-4 is a list of hash codes, the addresses of the e-mail mass mailer that were hashed to form LIST-1, are correlated or cross-referenced to addresses in LIST-4 in order to extract addresses to which an e-mail message may be sent. In step 324, LIST-4 is returned to the e-mail mass mailer 212 whereat LIST-4 is used to identify e-mail addresses in LIST-1 from which the elements of LIST-4 were calculated. Those e-mail addresses, the hash codes of which are in LIST-4, are addresses to which a commercial e-mail message may be sent.

[0067] In an alternate embodiment, the data network transmission time required to send LIST-4 back to the e-mail mass mailer 212 can be significantly reduced by subtracting hash codes that make up LIST-4 from LIST-1, to yield LIST-4A. In such an alternate embodiment, in step 323, LIST-4 is subtracted from LIST-1 to yield LIST-4A. LIST-4A is then a list of hash codes of LIST-1 that were purged by the list manager 210.

[0068] In most instances, LIST-4A will be considerably smaller than LIST-4. In step 325, LIST-4A, which is a list of hash codes of addresses that were purged, is uploaded to the e-mail mass mailer 212. Upon receipt of LIST-4A, the e-mail mail mass mailer subtracts LIST-4A from LIST-1 to yield LIST-4, albeit by the e-mail mass mailer 212.

[0069] In step 326, the e-mail mass mailer (212) extracts, correlates or otherwise recovers e-mail address strings, which when they are hashed, render values that are elements of LIST-4. If LIST-4A was sent to the e-mail mass mailer 212 in step 325, LIST-4A hash codes can be correlated with addresses that were hashed to make LIST-1 such that addresses that were hashed to yield an entry in LIST-4A are not sent an e-mail message.

[0070] In step 328, the e-mail mass mailer (212) broadcasts a commercial message to addresses that were extracted from the hash codes that make up LIST-4.

[0071] In an alternate embodiment, the list manager 210 can compile LIST-1 from other sources, such as the aforementioned web sites that collect information from prospective customers of the merchant 203. In such an embodiment where LIST-1 is compiled by the list manager 210, step 308 is not required. Step 312 would be modified to show that the e-mail list manager 210 was in receipt of LIST-1 and had previously calculated a hash code for each address to which an e-mail message could be sent.

[0072] In such an alternate embodiment, the list manager 210 could also accumulate an unsubscribe list and a black list using the same methods described above. The hash codes that make up LIST-3 would be calculated in step 318.Fig. 4 depicts step 322 of Fig. 3 in detail and shows a method of removing the hash codes of LIST-1 that also appear in either LIST-2 or in LIST-3.

[0073] In step 401, three loop counters "X," "Y," and "Z" are initialized to the value "1." The loop counters are used as pointers to elements of sorted LIST-1, LIST-2 and LIST-3. Loop counter *x* identifies a hash codes in LIST-1; loop counter *y* identifies hash codes in LIST-2. Loop counter *z* identifies hash codes in LIST-3, although LIST-3 can have hash codes that represent black listed e-mails and also unsubscribe list e-mails. In Fig. 4, when $x = 1$, the first value of LIST-1 will be read and compared to a *y*'th value from LIST-2. When $x = 2$, the second value of LIST-1 will be read and compared to an *x*'th value of LIST-1. When $y = 1$, the first value of LIST-2 will be read. When $z = 2500$, the 2500'th value of LIST-3 will be read.

[0074] In step 402, LIST-1, LIST-2 and LIST-3 are sorted using the aforementioned Quick Sort algorithm. In a preferred embodiment, LIST-1, LIST-2 and LIST-3 are sorted in ascending alpha-numeric order. When the lists are sorted in ascending alpha-numeric order, the first entry in each list will have the lowest alpha-numeric value of all values in each list. The last entry of each list will be the highest-valued entry in each list. When the lists are so ordered, the search for common hash codes can be accomplished faster by reading a first value from LIST-1. LIST-2 (or LIST-3) is then read, element-by-element. Each element from LIST-2 (or LIST-3) is compared to the first element of LIST-1 until a match between two element is determined or until the value of the LIST-2 hash code exceeds the value of the first LIST-1 hash code.

[0075] When the value of a LIST-2 hash code exceeds the value of the LIST-1 hash code, the next element of LIST-1 is read and compared to the preceding value from LIST-2 and all successive entries from LIST-2 until a LIST-2 value matches the LIST-1 value or until the LIST-2 value exceeds the LIST-1 value.

[0076] The comparison of LIST-1 entries to LIST-2 (and then LIST-3 entries) continues until the end of LIST-1 or LIST-2 is reached. When the end of LIST-2 is reached, LIST-3 is compared to LIST-1 in the same manner as was LIST-2.

[0077] The steps of the sorting methodology might not need to be performed if there are no elements in LIST-2 and LIST-3 that are in LIST-1. In order to avoid wasting processor time, in step 403, the lists, i.e., LIST-1, LIST-2 and LIST-3 (actually the two or more lists that comprise "LIST-3) are tested to determine whether the lists are mutually exclusive, i.e., whether there are no elements in any of the lists that are common. List mutual exclusivity is quickly and easily determined by comparing the end values of the lists, albeit after the lists are sorted.

[0078] One test of mutual exclusivity is performed by testing the value of the first element of LIST-1 against the value of the last element of LIST-2 and LIST-3. If the value of the first element of LIST-1 is greater in value that the value of the last element of LIST-2 and LIST-3, all of the elements of LIST-1 are greater in value than the elements of the other lists and there are no elements in LIST-2 or LIST-3 in LIST-1. If either LIST-2 or LIST-3 has a terminal value greater than the first value of LIST-1, there might be an element of LIST-2 or LIST-3 in LIST-1 so the steps after step 403 need to be performed as set forth below.

[0079] Another equivalent test for mutual exclusivity is testing the value of the last element of LIST-1 against the value of the first element of LIST-2 and LIST-3. If the last element of LIST-1 is smaller than the value of the first element of LIST-2 and LIST-3, there are no elements in LIST-2 or LIST-3 in LIST-1. If either LIST-2 or LIST-3 have an initial value less than the terminal value of LIST-1, there might be a value in LIST-2 or LIST-3 that exists in LIST-1 so the steps after step 403 need to be performed as set forth below. In Fig. 4, at step 404, using the loop counter *x* that was initialized to 1 in step 401 as a pointer, the first hash code from LIST-1 is read from memory into an accumulator or other register where it can be evaluated. In step 406, using the loop counter *y* as a pointer, the first hash code from LIST-2 is also read into a register from memory.

[0080] Inasmuch as both LIST-1 and LIST-2 entries are hash codes of the same length and since both LIST-1 and LIST-2 were sorted according to the same criteria, if the first hash code of LIST-1 is found in LIST-2, such an entry of LIST-1 is a hash code of an e-mail address to which a message should not be sent. Such an entry of LIST-1 is purged from LIST-1.

[0081] In step 408, a test is performed to determine whether the *x*'th hash code in LIST-1 is equal to the *y*'th hash code of LIST-2. If the result of step 408 is positive, the *x*'th hash code of LIST-1 is "purged" from LIST-1 in step 410, thereby creating at least part of LIST-4 by deleting an entry from LIST-1. After step 410, the loop counter *x* is incremented in step 414 by which the next hash code in LIST-1 will be tested against the *y*'th value of LIST-2, if the end of LIST-1 has not been reached as determined at step 418.

[0082] In step 418, a test is performed to determine if the LIST-1 pointer value of *x* (which was incremented in step 414) has reached a value *n*, which is shown in the legend of Fig. 4 to be the number of entries in LIST-1. If at step 418 *x* is not equal to *n*, the end of LIST-1 was not reached and program control returns to step 404 where the next *x* entry is read from LIST-1.

[0083] If the next *x* entry of LIST-1 equals the same *y* entry of LIST-2, it too is purged from LIST-1 in step 410. If not, program control proceeds to step 412.

[0084] If the result of the test at step 408 is negative, the *x* loop counter is not incremented as in step 414, but a test is instead performed at step 412 to determine whether the *x*'th entry is less than the *y*'th entry. If the result of the test at step 412 is positive, the *x*'th entry is smaller than the *y*'th entry and as a result, the next *x*'th entry is retrieved for testing by incrementing the *x* loop counter in step 414. Whether the end of LIST-1 has been reached is tested in step 418 as before.

[0085] If the test at step 412 is negative, the *x*'th entry of LIST-1 is greater than the *y*'th entry of LIST-2, which means the next *y* entry of LIST-2 should be tested against the current *x* entry of LIST-1. In step 416; the *y* loop counter will be incremented.

[0086] Step 420 tests whether the end of LIST-2 has been reached. If the result of step 420 is negative, program control returns to step 404 whereupon the next *y* entry is retrieved/tested from LIST-2. If at step 420 the result is positive, LIST-2 has been exhausted, at step 426, the *x* value is re-initialized and the *z* counter is initialized enabling the testing of LIST-3 against LIST-1.

[0087] As can be seen in steps 404 and 406, they are executed only if their respective values were changed in previously-executed steps.

[0088] It should be apparent to those of ordinary skill that, when program steps 404, 406, 408, 410, 414 and 418 are executed, every occurrence of an entry from LIST-2 in LIST-1 causes the deletion or purging of the LIST-2 entry from every occurrence of LIST-1. Inasmuch as the lists were sorted and may contain multiple entries of a hash code, repeated occurrences of an e-mail in LIST-1 that appears in LIST-2 are all purged.

[0089] With respect to the test at step 408, if an *x*'th value of LIST-1 is not equal to an *y*'th value of LIST-2, a test is performed in step 412 to determine if the *x*'th hash code value being compared to the *y*'th value, is less than the *y*'th value. Because the lists LIST-1, LIST-2 and LIST-3 were sorted in ascending order, a determination that the *x*'th value is less than a *y*'th value at step 412 means three things: first, the two hash codes are not identical; second, the *x*'th hash code does need to be purged; third, the next *x*'th value needs to be tested against the same *y*'th value before incrementing *y* in order to determine whether the next *x* value might be in LIST-2.

[0090] If at step 412 the *y*'th value is greater than the *x*'th value, at step 414, *x* is incremented again in step 414 in order to retrieve the next-highest valued hash code in LIST-1 and compare it to the previously tested *y* value.

[0091] If at step 412, the *x*'th value is *not* less than the *y*'th value, the x'th value is therefore *larger* than the *y*'th value (because the test at step 408 failed). The loop counter *x* is not incremented. The *y* loop counter is incremented instead at step 416 in order to retrieve the next-greater valued hash code in LIST-2. Before retrieving the next hash code from LIST-2, at step 420 the *y* loop counter value is checked against the value of *m*, which is set equal to the number of entries in LIST-2, as shown in Fig. 4.

[0092] If the value of *y* at step 420, is less than *m*, meaning that the last entry of LIST-2 has not been checked, program control returns to step 404 where the previously-read *x* value is read at step 404 followed by the retrieval of the next value from LIST-2. At step 408, the value of the *x*'th and *y*'th values are compared as before.

[0093] It should be apparent to those of ordinary skill that program steps 404, 406, 408, 412, 416 and 420 compare a particular hash code from LIST-1 and to hash codes from LIST-2. These steps check the particular hash code from LIST-1 against successive values from LIST-2 until a hash code of LIST-2 value equals the *x* value or, until a *y* value greater than the *x* value is read, or until the end of LIST-2 is reached.

[0094] Step 426 is reached when the end of LIST-1 or LIST-2 has been reached. At step 426, the LIST-1 loop counter *x* is reset to 1 and a LIST-3 loop counter, *z* is also set to 1 so that LIST-3 can be checked against LIST-1.

[0095] At step 430, entries of LIST-1 are compared against entries of LIST-3 using the same search strategy set forth above. Inasmuch as LIST-3 is described as possibly including a black list and/or an unsubscribe list, LIST-3 can actually include two (2) lists. In such a case, the steps 430-450 need to be repeated for both the unsubscribe list and the black list. In an alternative embodiment, the unsubscribe list entries and the black list entries can be merged into a single list that is compared to LIST-1 entries. For purposes of illustration however, only one component list of LIST-3 is depicted as being checked against entries of LIST-1.

[0096] At step 430, the first entry of LIST-1 is read. At step 432 the first list of LIST-3 is read.

[0097] At step 434, the *x*'th entry of LIST-1 is compared to the *z*'th entry of LIST-3. If these two hash codes are identical, program control proceeds to step 438, whereat the *x*'th entry of LIST-1 is purged because it was detected in step 434 as being equal to the *z*'th entry of LIST-3.

[0098] If at step 434 it is determined that the *x*'th entry of LIST-1 is not equal to the *z*'th entry of LIST-3, the test at step 436 determines if the *x*'th entry value of LIST-1 is less than or greater than the *z*'th entry of LIST-3. Inasmuch as LIST-1 and LIST-3 are sorted in ascending order, if the *x*'th entry of LIST-1 is larger than the *z*'th entry of LIST-3, the next entry of LIST-3 is read by incrementing the *z* loop counter at step 440. At step 444, the z loop counter is checked against the length of the LIST-3 to determine if the last entry of LIST-3 was checked. If yes, the loop terminates at step 448. If no, program control returns to step 430 where the previously-read value of LIST-1 is compared against the next value of LIST-3.

[0099] In light of the foregoing, it should be apparent to those or ordinary skill that the method depicted in Fig. 4 provides for a comparison of hash codes in LIST-1 against hash codes in LIST-2 and LIST-3. Each hash code of LIST-1 is checked against a series of hash codes from LIST-2 and LIST-3 until the value from LIST-2 and LIST-3 exceed the hash code of LIST-1 being tested whereupon the next-highest hash code from LIST-1 is checked against the most-recent value from LIST-2 and LIST-3.

[00100] In a preferred embodiment, the hash codes were forty characters in length.

[00101] By Quick sorting each list and checking values of LIST-1 against values of LIST-2 and LIST-3, it is possible to quickly sort through millions of hash codes in the lists to purge hash codes from LIST-1 that appear anywhere in LIST-2 or LIST-3.

[00102] In an alternate embodiment, entries of LIST-2 and LIST-3 can be hash codes of e-mail addresses to which commercial messages *can* be sent instead of hash codes of e-mail addresses to which e-mail messages should *not* be sent. In such an alternate embodiment, hash codes in LIST-1 that do *not* also appear in LIST-2 or LIST-3 should be purged from LIST-1, from which LIST-4 is created, instead of being purged from LIST-1 if they *do* appear in LIST-2 or LIST-3.

[00103] In yet another embodiment described above, LIST-4 hash codes can be subtracted from the LIST-1 hash codes to yield a "LIST-4A." In such an embodiment, LIST-4A represents the addresses of LIST-1 that appear in either LIST-2 or LIST-3 and which were purged from LIST-1. Inasmuch as LIST-4A is usually much shorter than LIST-4, calculating LIST-4A means that the time required to transmit hash codes to purge can be significantly reduced. Upon receipt of LIST-4A, the e-mail mass mailer 212 purges LIST-4A entries from LIST-1 yielding LIST-4. Entries of LIST-4 can be correlated with the original e-mail lists from which LIST-1 hash codes were calculated to yield the list of e-mail targeted addresses, identified in Fig. 2 by reference numeral 224.

[00104] Although the preferred embodiment contemplates that the function of the e-mail list manager 210 is separate and apart from both the e-mail mass mailer 212 and the commercial advertiser's computer 202, the steps of computing and comparing hash codes of the commercial advertiser's suppression list (or authorization list) to the hash codes of an e-mail mass mailer 212 could all be performed by the merchant's computer 202. Conversely, the hash code computation and comparisons could all be performed by the e-mail mass mailer. Accordingly, the salient aspects of the method are the reception of a first set of hash codes, each of which represent an e-mail address to which a message can be sent. The entries of the first set of hash codes are compared to entries in at least a second set of hash codes, each of which represents an address to which an e-mail address should *not* be sent. Hash codes in the first set that are also in the second set are purged from the first set, leaving a set of hash codes, each of which represents an address to which a commercial e-mail message may be sent.